School of Computer Science, University of Nottingham

G51PGP Programming Paradigms, Spring 2019

Graham Hutton

Haskell Coursework I, Part 1/3

Deadline: Wednesday 6th February 2019, 12 noon

This exercise sheet is worth 1% of the overall module mark. It is assessed on a simple pass/fail basis: if you complete the sheet you receive full marks, otherwise no marks. You should attempt to complete the exercises in your own time, but if you get stuck you can ask for help from the tutors during the weekly lab session.

Assessment will be carried out by oral examination during the lab session – nothing needs to be handed in. When you have completed the exercises ask a tutor to examine your solution. The tutor will then ask you some questions to test your understanding. Note that tutors will only be available to answer questions and assess your solution during the official G51PGP lab sessions (Wednesdays, 11am to 1pm, A32).

The Glasgow Haskell Compiler (GHC) provides an interactive interpreter (GHCi), which will be the main Haskell tool used in this module. The usual way to write Haskell programs is to have two windows open: one for a text editor to write your code, and the other for GHCi so that you can regularly load and test your code.

1. Open a text editor. On the lab machines I recommend using Notepad.

2. Type in the following function definition:

   ```
   double x = x + x
   ```

3. Save your file as

   ```
   script1.hs
   ```

4. Load your file into GHCi. If you're using Windows on the lab machines, double-clicking on your file will open GHCi and load the file, provided the filename ends in `.hs`. If you're using a command line interface, e.g. on the School's Linux servers, then navigate to the directory containing your file and type:

   ```
   ghci script1.hs
   ```

   In either case, GHCi should load and you should see something like this:

   ```
   [1 of 1] Compiling Main             ( script1.hs, interpreted )
   Ok, modules loaded: Main.
   *Main>
   ```

5. Test your function on some numbers. For example, type:

   ```
   double 7
   ```

6. Add the following to `script1.hs` on a new line, and resave the file:

   ```
   quadruple x = double (double x)
   ```

7. The file can then be reloaded into GHCi using the command `:reload`, which can be abbreviated by `:r`. Now test the `quadruple` function on some numbers.

8. Add the following functions to your file, and then test them in GHCi:

   ```
   smallest x y = if x < y then x else y
   largest x y = if x > y then x else y
   ```

   Don't forget to reload before testing, and to resave before reloading!

9. Tab characters can cause problems in Haskell, as indentation is used to indicate grouping of definitions, but different editors interpret tabs in different ways. **The best way to avoid problems is not to use tabs in Haskell programs.**

   Now add and test the following code, making sure that the `l` and `s` after the `where` line up in the same column, which indicates that they are both local definitions:

   ```
   diff x y = l - s
              where
                 l = largest x y
                 s = smallest x y
   ```